EUROPA Coding Practices

General Practices

- Ensure you declare variables and methods in their narrowest scope.
- If you declare a static variable inside a non-static method, double check that the method should not be static and also double check that the variable should not be a member of the class.
- We discourage writing code in header files unless needed for templates or proven performance.
- Use STL classes and methods unless what you need is not provided. Same goes for any other code. Re * use as much as possible.

Pre-processing

- Include system headers by using the angle bracket style. (#include <stdio>)
- Include user files by using the double quote style. (#include "File.h")
- Do not define your own pre-processor macros to control level of or presence of debugging output or error checks.

Namespaces

- Use the std:: prefix, or 'using namespace std;' when using STL.
- Put Europa code in the Europa namespace.

Constants

- Use DEFINE_GLOBAL_CONSTANT and DECLARE_GLOBAL_CONSTANT for globals.

Class Members

- When handling static data, you must provide an automatic purge mechanism or provide an explicit purge method.

Initialization and Termination

- We should standardize method calls to initialization and termination methods. Such as nddl initialization which cascades onto constraint engine initialization.

Use

- Use const iterators unless you have to use a non-const iterator.
- When using const iterators, use ++iterator rather than iterator++.

References

- Direct pointer references are discouraged; use class Id instead.
- When creating a reference, create an m member that holds the id that gets constructed in the constructor initializer, in the destructor the m should be removed.

- When deleting references to ids call delete on the cast operator (e.g. delete (ConstrainedVariable? *) ref).

Numbers

- Define an enumerated type to handle number references instead of using magic numbers.

Classes

- Capitalize names of classes. When composing names for classes capitalize the first letter of each word.
- Declare a virtual destructor.

Virtual Classes

- Declare a protected constructor.
- Declare all functions pure virtual.

Methods

- Declare a method const where possible.
- Do not return bare pointers or non-const references.
- If the caller can own a data structure that is to be populated in the callee, create the data structure in the caller and then pass it by reference as an argument.
- Avoid copying of data structures where possible.
- Declare non-primitive arguments as const references.
- Return non-primitive values as const references.

Checks

- Use checkError to express pre-conditions.
- Use checkError to express invariants.
- Use checkError to express post-conditions.
- Avoid using non-const functions in checkError tests.
- Do not use assert.
- Do not use Id::isValid outside of checkError.
- Do not write "if (Test) checkError(...Check...);". Write "checkError(!Test $$ ...Check...)".

Output

- Use the Europa debugging output management system.
- Do not put debugging output into stdout or stderr.

Documentation

- Use doxygen style comments with the javadoc style keywords.